An Adaptive Partitioning Scheme for DRAM-based Cache in Solid State Drives

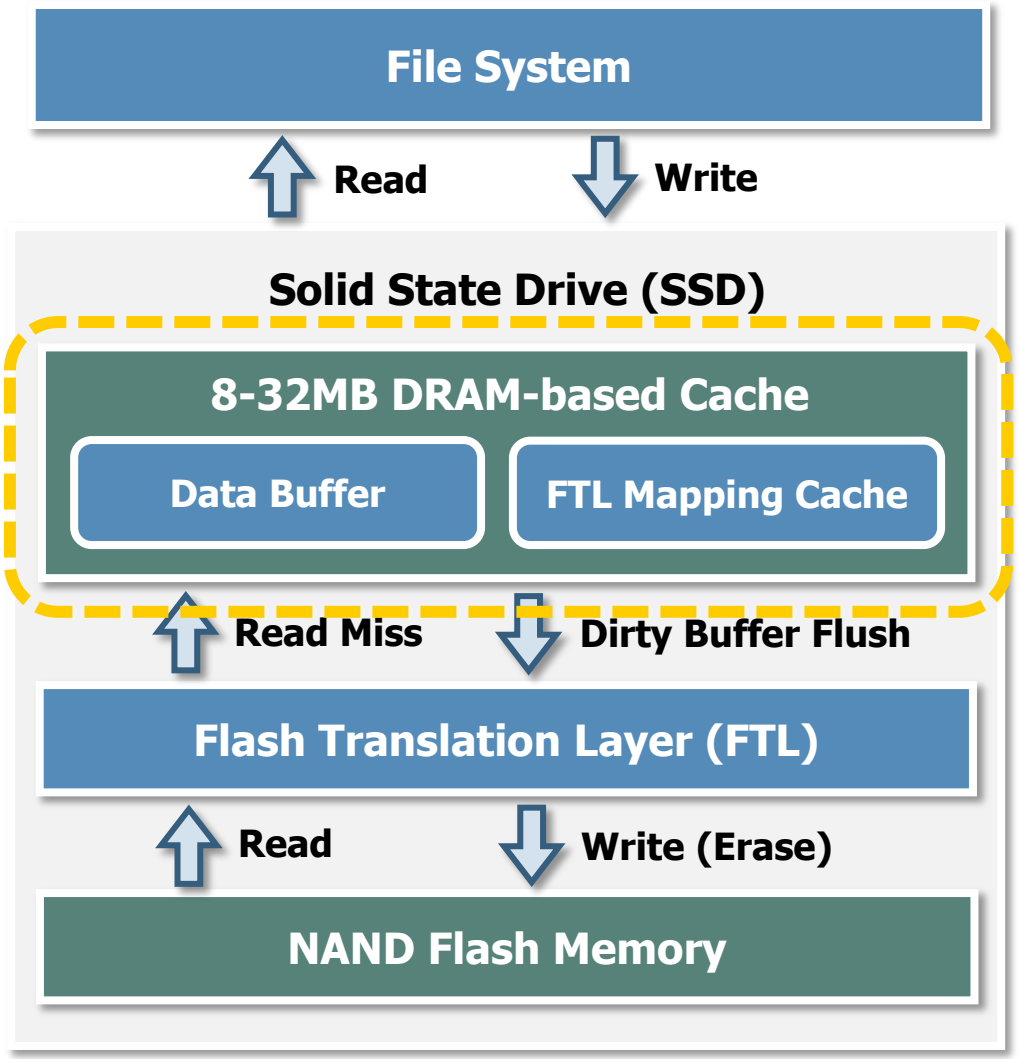
Hyotaek Shim, Bon-Keun Seo, Jin-Soo Kim, and Seungryoul Maeng

Korea Advanced Institute of Science and Technology (KAIST)
Sungkyunkwan University (SKKU)

IEEE MSST 2010 Research Track

Architecture of a Typical SSD

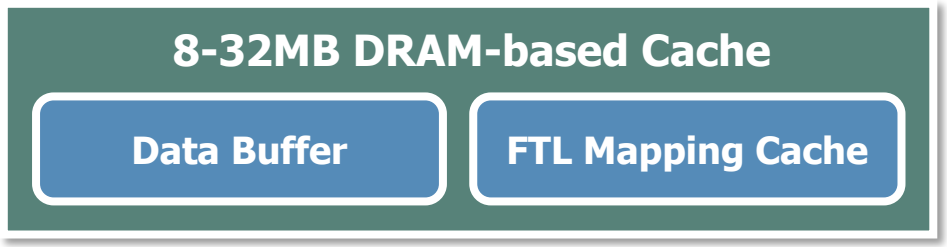
- The internal device cache has two main purposes
  - To absorb frequent read/write requests
  - To store logical-to-physical address mapping information
- We focus on how to efficiently utilize the device cache between the two purposes



Trade-Off between Buffering and Mapping

- A trade-off between how much space is allocated to buffering versus mapping
  - Frequent requests can be more cached with larger buffering space
  - The SSD performance can also benefit from larger mapping space
- The device cache should be appropriately partitioned
  - Existing studies assumed that the *BM ratio* is fixed (*static partitioning policy*)

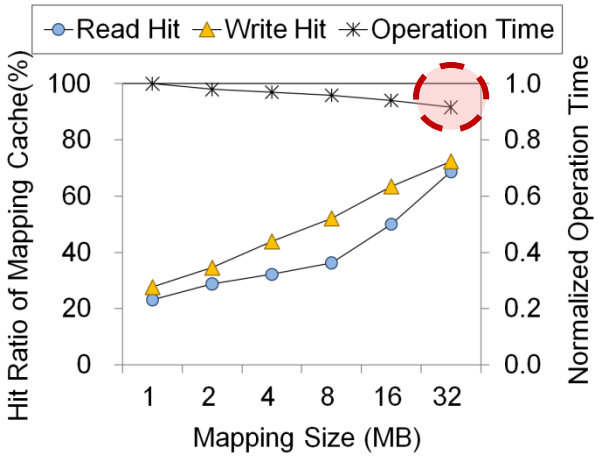
BM ratio is the ratio of the buffering and the mapping space.



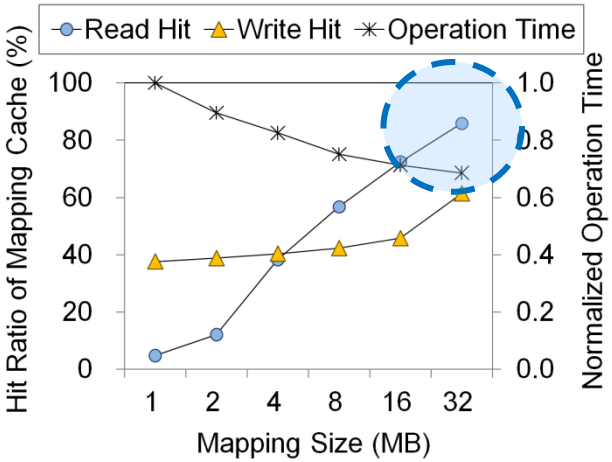
Effects of Adjusting the BM Ratio

- The optimal BM ratio is usually affected by workload characteristics

PC Trace

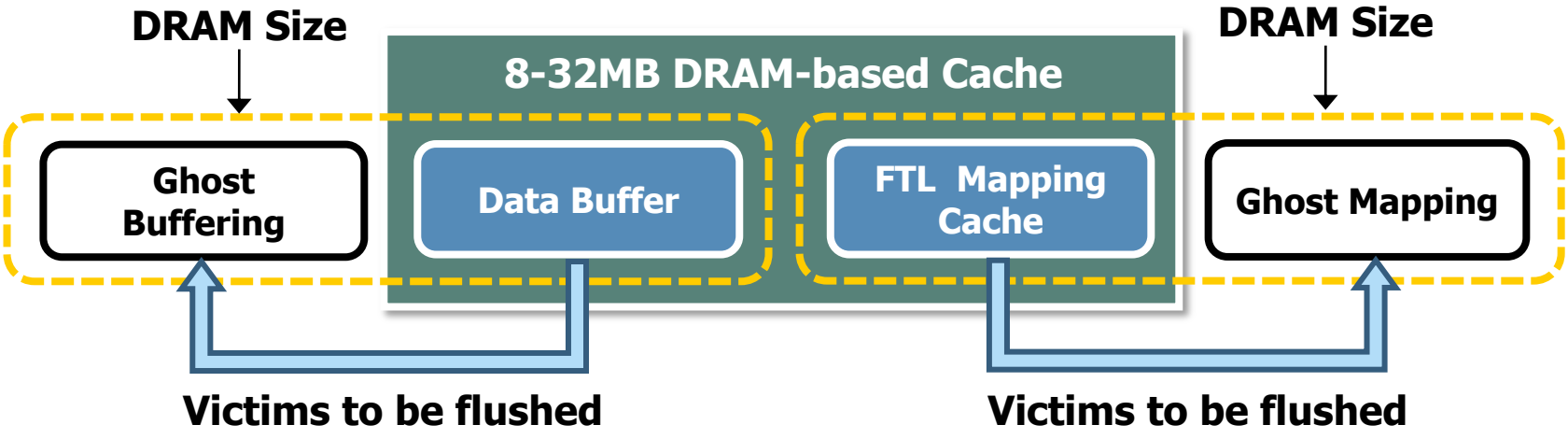


TPC-C Trace



Adaptive Partitioning Scheme (1/3)

- This scheme adaptively adjusts the BM ratio according to workload characteristics
  - Comparing the cost-benefits of buffering and mapping
  - Ghost cache
    - Exclusive victim cache that stores only metadata
    - The cost-benefits of actual caches are estimated by their ghost caches

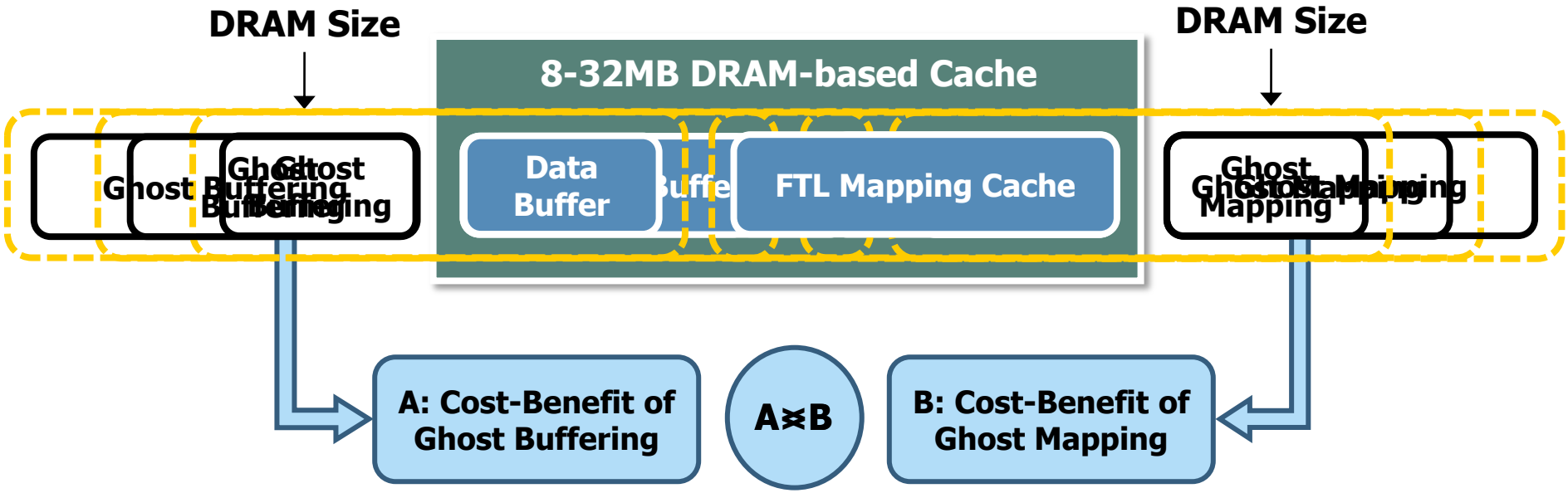


Adaptive Partitioning Scheme (2/3)

- *Cost-benefit* of ghost cache
  - Whenever a read/write hit occurs in ghost cache, its *benefit* is accumulated
    - At the same time, a read/write miss occurs in its actual cache
    - The *benefit* is the cost (NAND flash operation time) caused by the read/write miss in its actual cache
    - We call this cost opportunity cost caused by not enlarging the actual cache size
  - The *cost* of *cost-benefit* is the expected memory consumption of ghost cache

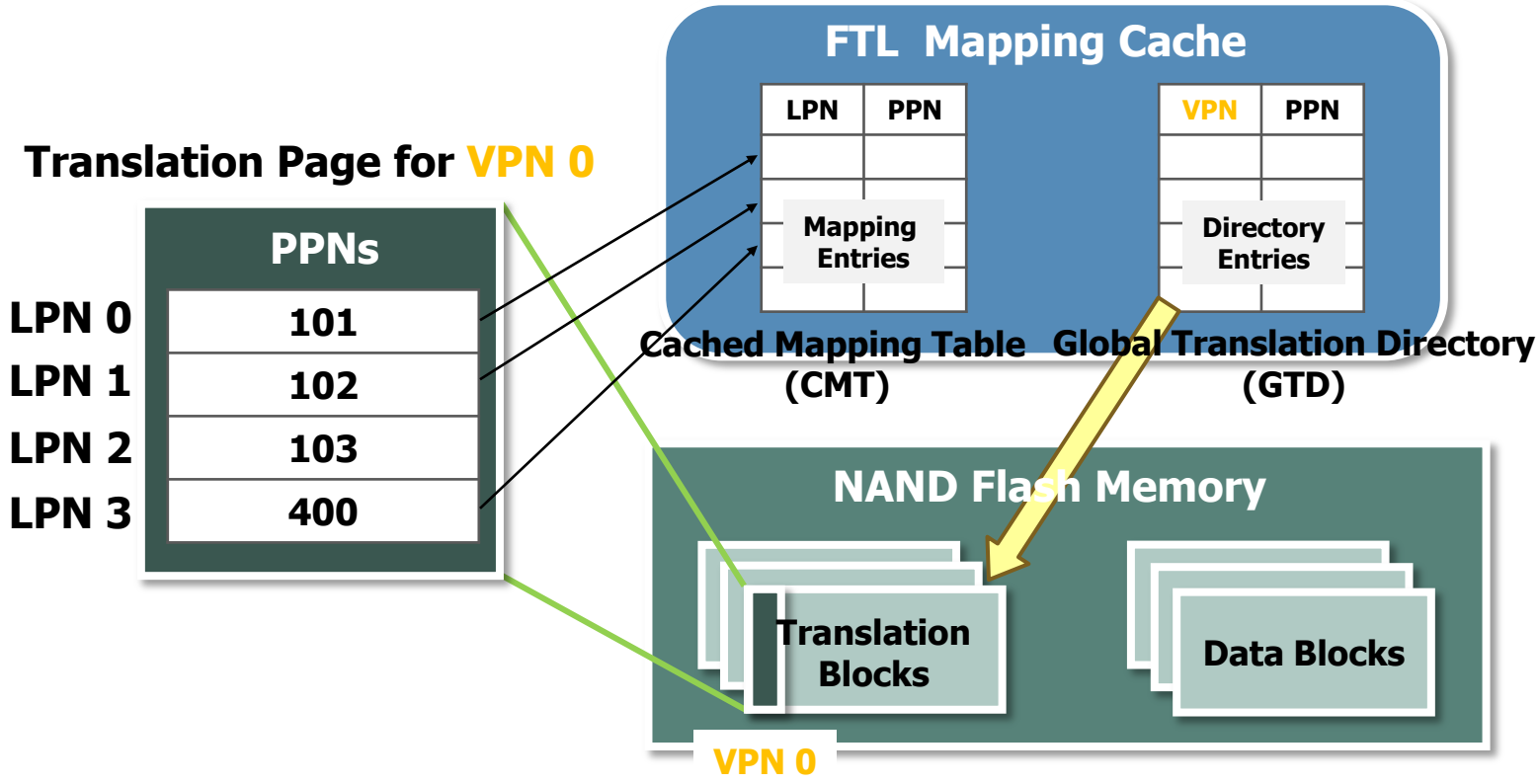
Adaptive Partitioning Scheme (3/3)

- At every pre-defined interval, the BM ratio is tuned by comparing the cost-benefits of ghost caches



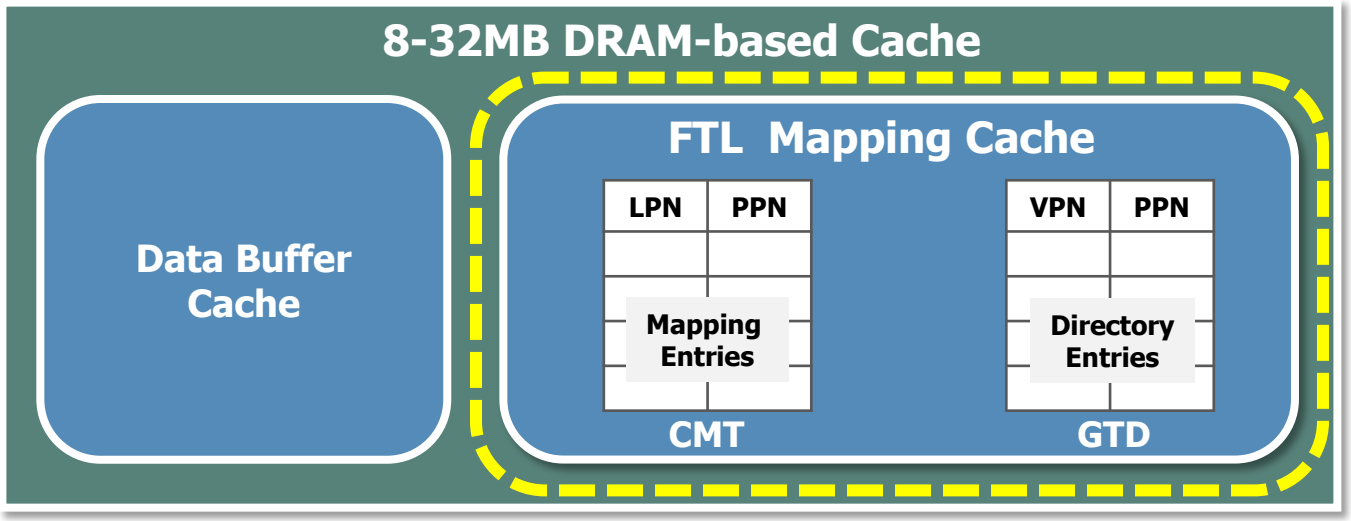
Case Study I
Demand-based Flash Translation Layer (DFTL)

- DFTL applies a caching mechanism to existing page-level mapping FTL
  - DFTL keeps only frequently-accessed logical-to-physical mapping entries in CMT



Opportunity Cost of the Mapping Cache with DFTL

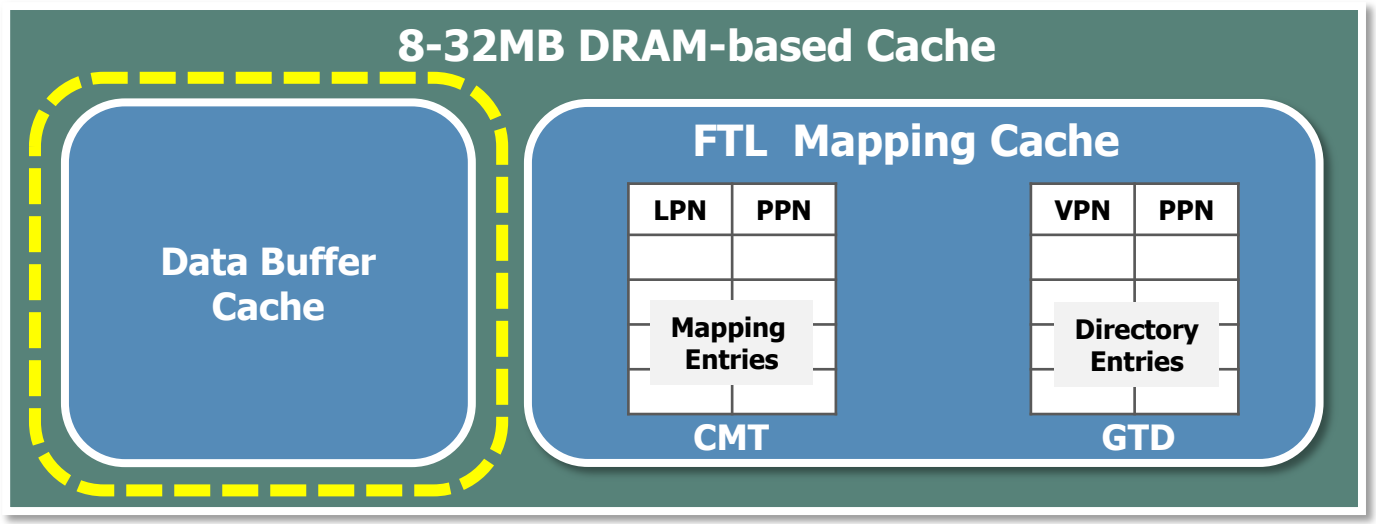
- Opportunity cost for a read miss in CMT
  - *Flash_Read*
- Opportunity cost for a write miss in CMT
  - {*Flash_Read* + *Flash_Write* (GC Overhead)} / *Batch_Factor*
  - *Batch_Factor* means the avg. # of CMT entries flushed by a batch update



Opportunity Cost of the Buffer Cache with DFTL

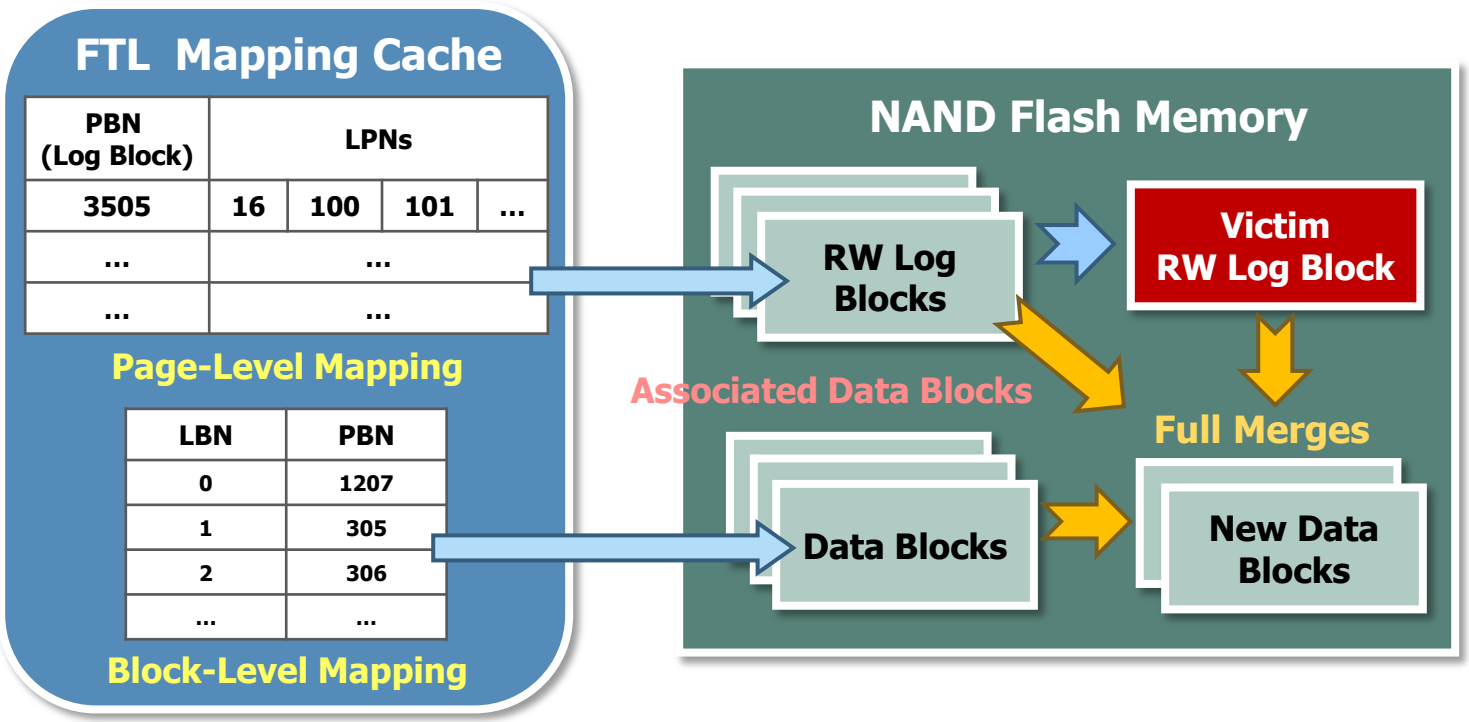
- **Opportunity cost for a read miss in buffer cache**
  - ***Flash_Read + Flash_Read * CMT_Read_Miss_Ratio***
- **Opportunity cost for a write miss in buffer cache**
  - ***Flash_Write (GC_Overhead)***

  + CMT_Write * CMT_Write_Miss_Ratio



Case Study II
Fully Associative Sector Translation (FAST)

- **Data blocks** are managed by block-level mapping
  - Where all pages must be fully and sequentially written
- A fixed number of **log blocks** handle updates



Case Study II
Fully Associative Sector Translation (FAST)

- **Write-dominant and high temporal-locality**
  - **Many valid pages in log blocks can be invalidated by following updates with enough log blocks**

Page-Level Mapping

PBN (Log Block)	LPNs							
3505	7	100	101	102	15	5	3	14
...				...				

If all valid pages of the same associated data block (LBN:12) are invalidated, a full merge is avoided

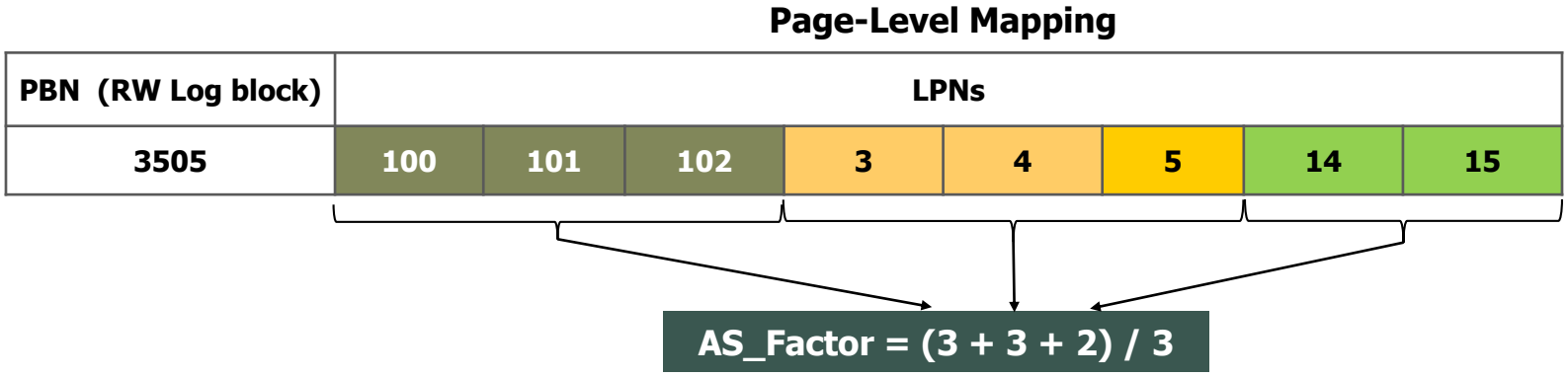
- **Read-dominant or small working set**
  - **Many log blocks remain unused, unnecessarily wasting the device cache**
  - **The unused mapping space can be utilized for buffering by reducing # of available log blocks**

Page-Level Mapping

PBN (Log Block)	LPNs			
3505	16	100	101	...
...	Empty			
...	Empty			

Opportunity Cost of the Mapping Cache with FAST

- **Opportunity cost for a read miss in mapping cache**
  - **There is no read miss in FAST**
- **Opportunity cost for a write miss in mapping cache**
  - ***Full_Merge_Overhead* / *AS_Factor***
  - ***AS_Factor* means the avg. # of written pages that belong to the same associated data block**



Opportunity Cost of the Buffer Cache with FAST

- **Opportunity cost for a read miss in buffer cache**
  - ***Flash_Read***
- **Opportunity cost for a write miss in buffer cache**
  - ***Flash_Write + RW_Log_Merge_Cost / #Pages_Per_Block***

Configurations for Experiments

- 8MB or 16MB of DRAM is assumed as the device cache

	DFTL	FAST
Tuning Interval	1,000 Requests	10,000 Requests
Tuning Unit Size	A CMT Entry	A Log Block

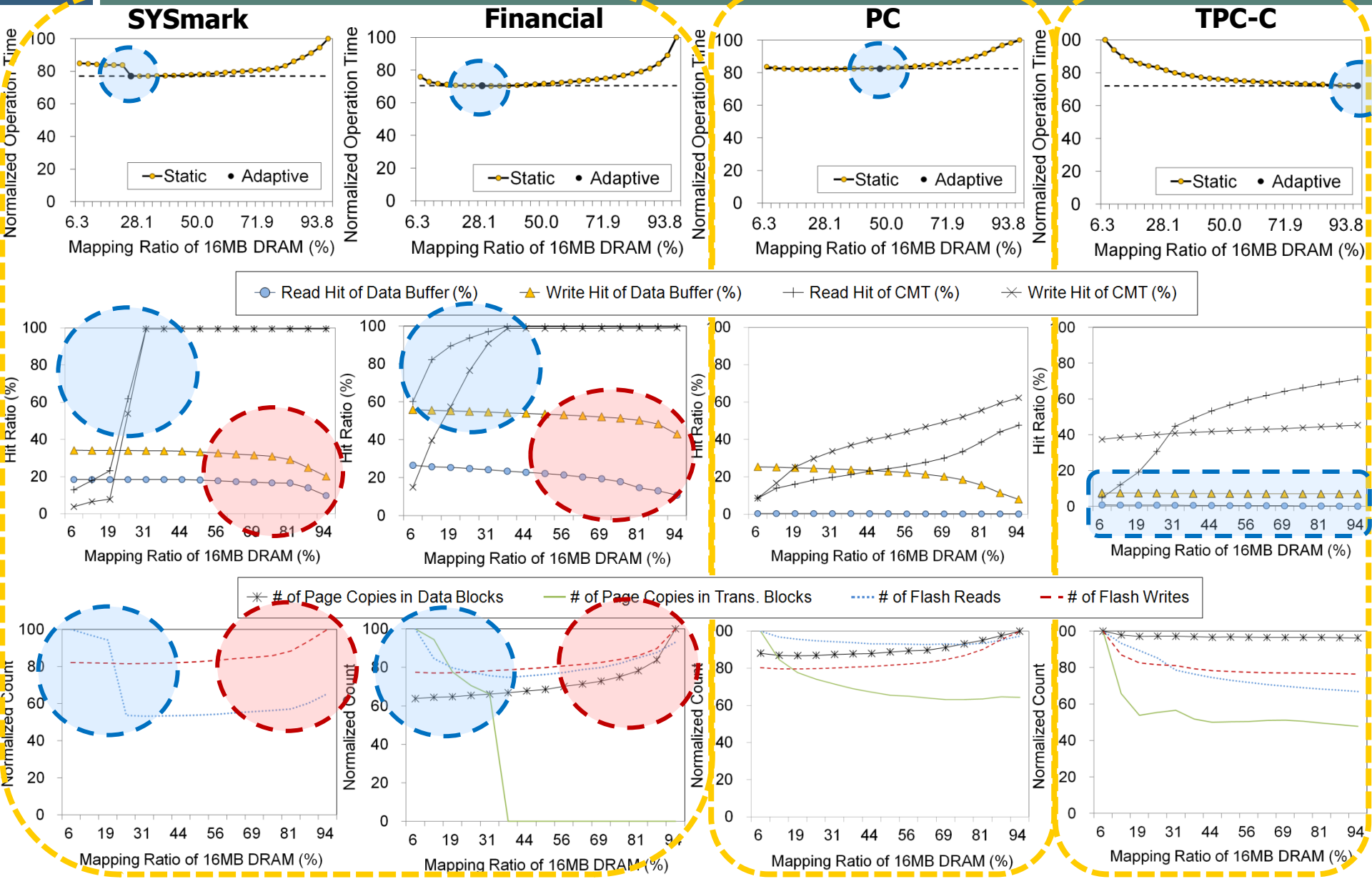
- 64GB SLC NAND flash memory
  - The number of extra blocks is set as up to about 10% of the total capacity

Flash Type	Unit Size (KB)		Access Time (μs)		
	Page	Block	Read	Write	Erase
SLC	2	128	72.8	252.8	1500

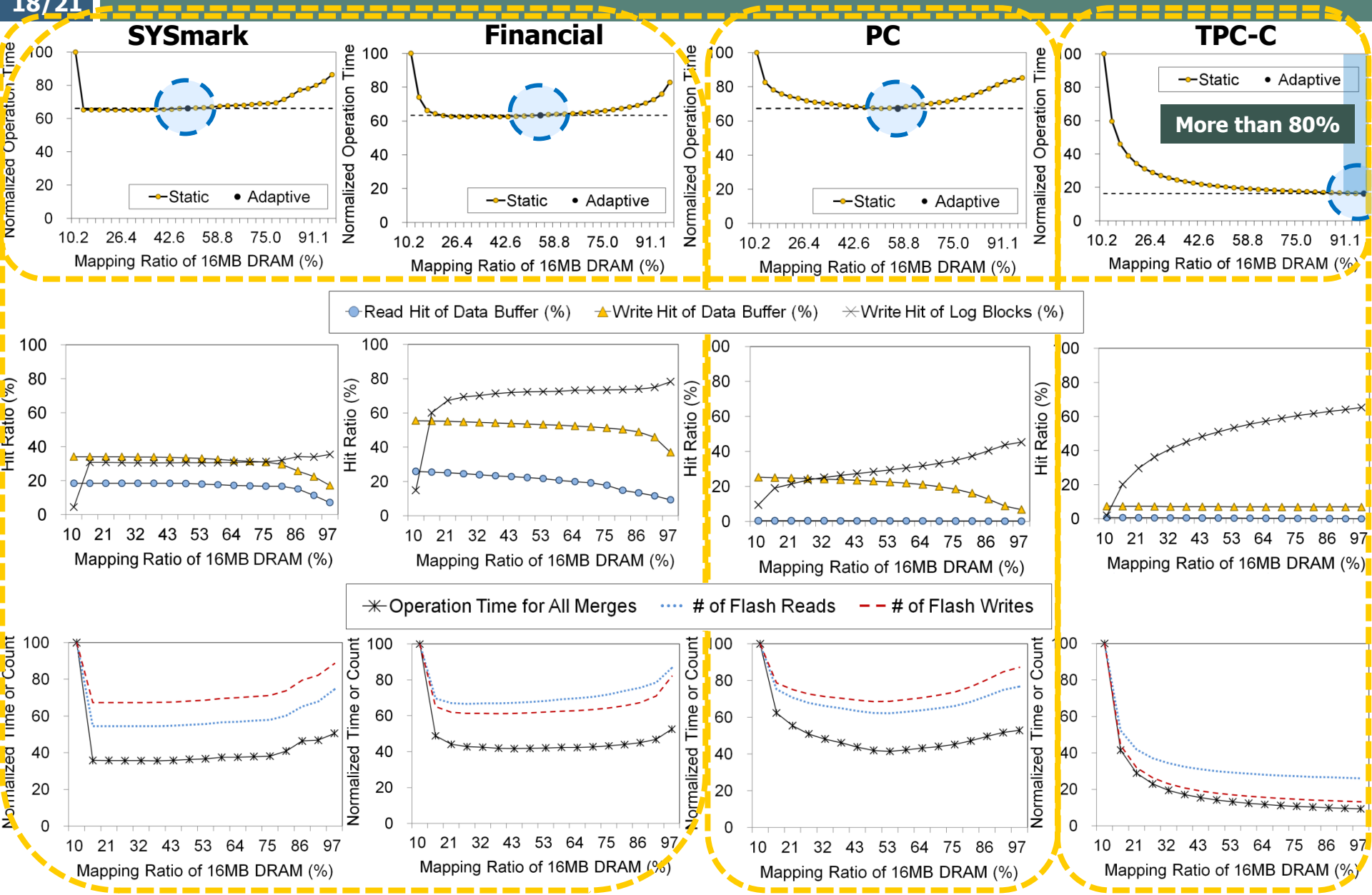
Summary of the Block-Level Traces

Name	Description	Avg. Req. Size [Read/Write] (KB)	Req. Ratio [Read/Write] (%)	Working Set [Read/Write] (GB)
SYSmark	Running *SYSmark 2007 Preview* including e-learning, office works, video creation, and 3D modeling	13.6 / 20	33 / 67	0.11 / 0.24
Financial	I/O trace from an OnLine Transaction Processing (OLTP) application running at a financial institution	2.3 / 3.6	47.4 / 52.6	0.45 / 0.5
PC	Document-based realistic workloads using various office applications	20 / 13.4	23.7 / 76.3	5.82 / 8.45
TPC-C	Running a TPC-C benchmark test with *Benchmark Factory*	2.2 / 2.1	81.4 / 18.6	8.04 / 4.45

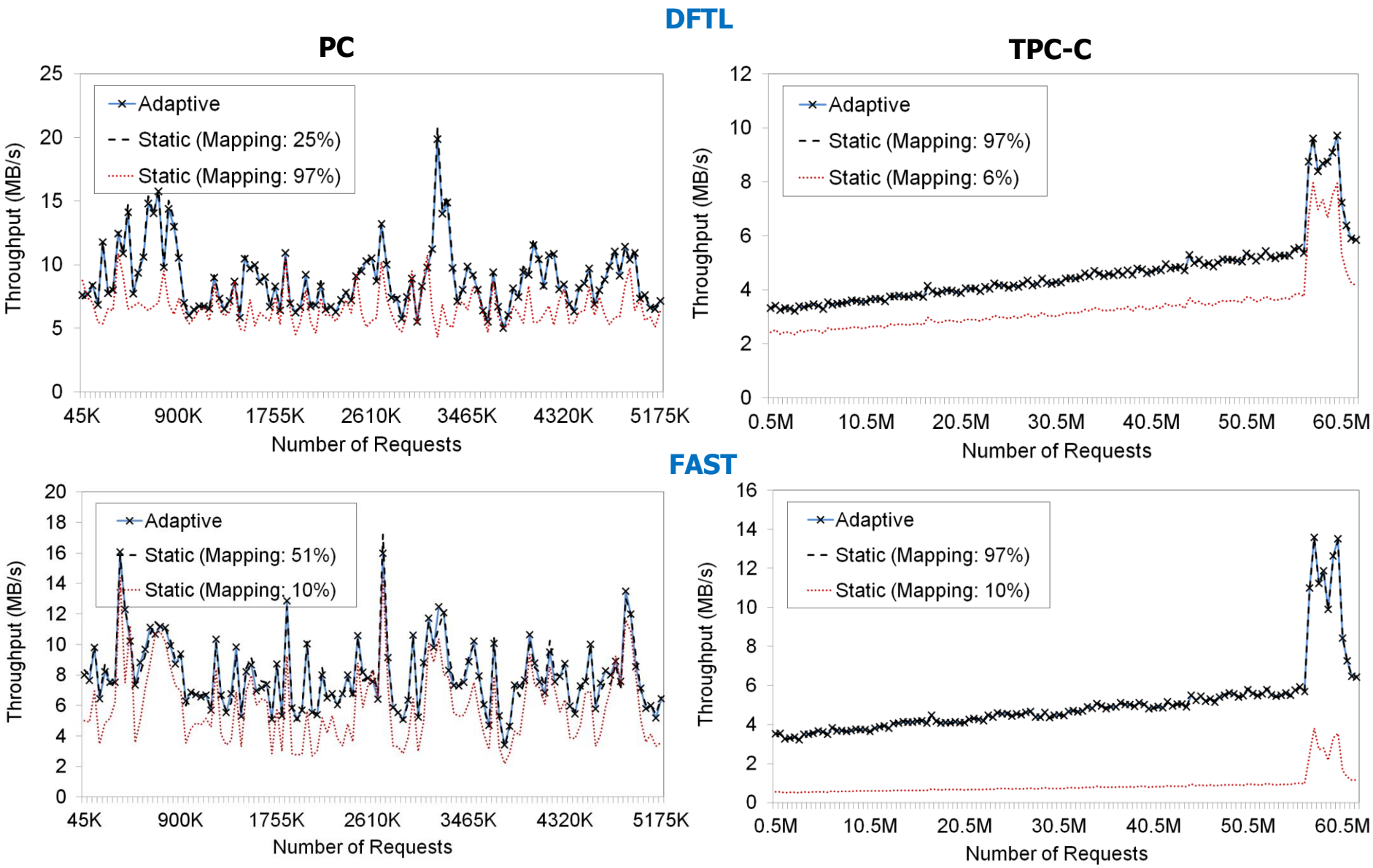
Operation Time with DFTL



Operation Time with FAST

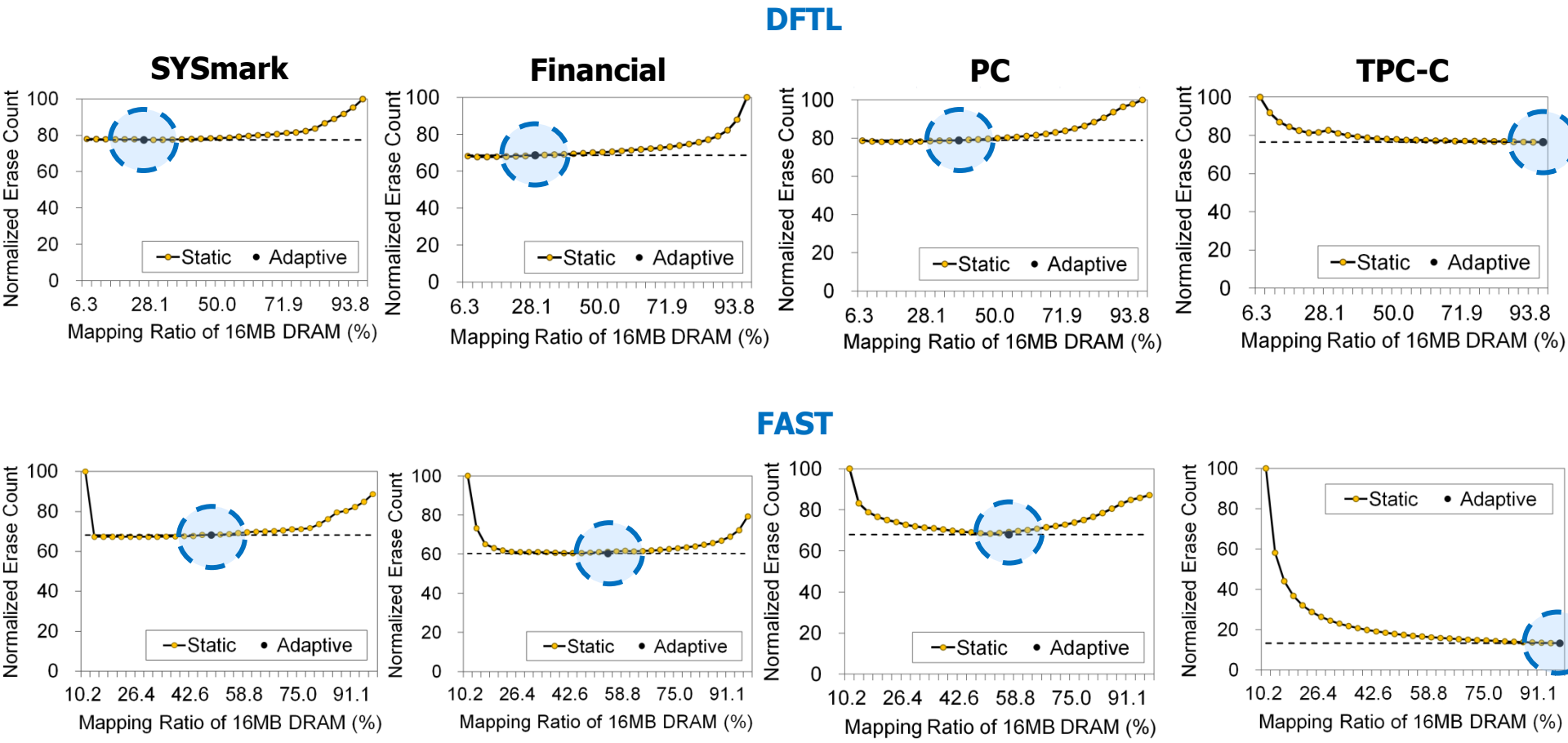


Throughput with DFTL and FAST



Erase Count with DFTL and FAST

- The proposed scheme can extend the lifetime of SSDs



Conclusions

- We proposed an adaptive partitioning scheme for better performance of SSDs
  - The proposed scheme adaptively tunes the BM ratio according to workload characteristics
  - We built a cost-benefit model based on a *ghost caching mechanism*
  - The performance results come near the best performance under the static partitioning policy with varied workloads
- We expect that SSDs equipped with the proposed scheme can be deployed in different environments without workload-specific tuning

Extra Slides

Implementation of Ghost Mapping Cache

- ***Bloom Filter***
  - **To insert an LPN, the corresponding hash bucket is set**
    - **If a hash collision occurs, the bit count of hash bitmap does not increase**
  - **To flush a victim, we reset of the bit in random order**
    - **the bitmap hash is flushed only if (the current bit count of the hash bitmap) *α is smaller than the bit count that should be preserved without collisions**
    - **α was set as 3 for all the simulation results**